AI复习（预习

by CurryWOE

评估函数(evaluation function)

评估函数是一种用于估计给定游戏状态值的启发式方法。

它用于确定哪个动作对给定的玩家来说是最佳的。

评估功能考虑了游戏的当前状态，例如棋盘上的棋子、棋子的位置以及每个玩家的物质优势。

评估函数通过为每个可能的移动提供分数来处理alpha beta修剪。

最小最大搜索树(minmax search tree)

表示双人博弈的所有局面的数据结构。

它是一颗树，两个节点有边说明能从一个局面走向其后续局面。

叶节点有一个评估函数值，评估函数值越大，先手赢的越多。

双人博弈是先后手交替进行。

默认，根节点是先手操作，从左到右搜索。

Alpha-Beta剪枝(Alpha-Beta pruning)

因为双方都想赢，先手会选择最终结果评估值大的后续局面；

反之，后手会选择最终结果评估值小的后续局面。

这里指的最好最坏结果指的都是双方积极操作的结果。

当先手操作时，设此时要走向节点 v，如果走向 v之后，后手能使最终结果比之前最坏结果还坏，那么先手不会选择走向 v；

反之，当后手操作时，设此时要走向节点 v，如果走向 v之后，先手能使最终结果比之前最坏结果还坏，那么后手不会选择走向 v

例题1

先手从A走到B，后手为了使最终结果小，后手会选择5，所以先手从A走到B的最坏结果是5

先手从A走到C，后手为了使最终结果小，后手会选择4，这比之前最坏结果5还坏，所以停止搜索C

先手从A走到D，后手为了使最终结果小，后手会选择2，这比之前最坏结果5还坏，所以停止搜索D

A的最坏最终结果是5，所以第一个填5

B的最坏最终结果是5，所以第二个填5

D的最坏最终结果是2，所以第三个填2

根据剪枝，搜索了（5，7，6，8，4，2）共6个叶节点

曼哈顿距离(Manhattan distance)

横坐标差的绝对值，纵坐标差的绝对值，的和

例题2

$|10-20|+|5-(-5)|=20$

根据期望决定最佳行动(determine the best action)

对于每种行动，计算结果的期望，选取期望最大的结果对应的行动

信息价值(value of information)

信息价值=有信息的收益-没信息的收益

当没信息时，默认各种情况等概率

又叫VPI，虽然不知道P是哪里来的

VPI的属性：

非负性，不可加性，顺序独立性（我理解为交换律结合律）

例题3

200元单车，每年折旧50元，每年花费30元可以换新，5%的概率被偷每年，决定最佳行动，计算“5%的概率被偷每年”的信息价值

不换新，期望$=-50*0.95+-200*0.05=-57.5$

换新，期望$=-30*0.95+(-200-30)*0.05=-40$

$-40>-57.5$，所以选择换新

没信息时，

不换新，期望$=-50*0.5+-200*0.5=-125$

换新，期望$=-30*0.5+(-200-30)*0.5=-130$

$-125>-130$，所以选择不换新

信息价值$=-40-(-125)=85$

启发函数(heuristic)

自定义函数，定义的好坏影响算法的准确性和速度

提供从给定节点到目标节点的最便宜路径的成本估计

A*算法

$f = g + h$

g 是起点到该点的费用

h 是启发函数

当 $h = 0$，算法退化成Dijkstra算法

当 h 始终小于等于节点到终点的代价，能找到最短路。但是当 h 的值越小，算法将遍历越多的节点，也就导致算法越慢

当 h 完全等于节点到终点的代价，将找到最佳路径，并且速度很快。可惜的是，并非所有场景下都能做到这一点。因为在没有达到终点之前，我们很难确切算出距离终点还有多远

当 h 的值比节点到终点的代价要大，则不能保证找到最短路径，不过此时会很快

在另外一个极端情况下，如果 $h >> g$，则此时只有 h 产生效果，这也就变成了最佳优先搜索(Best First)

完整的A*算法描述如下：

```
* 初始化open_set和close_set；
* 将起点加入open_set中，并设置优先级为0（优先级最高）；
* 如果open_set不为空，则从open_set中选取优先级最高的节点n：
    * 如果节点n为终点，则：
        * 从终点开始逐步追踪parent节点，一直达到起点；
        * 返回找到的结果路径，算法结束；
    * 如果节点n不是终点，则：
        * 将节点n从open_set中删除，并加入close_set中；
        * 遍历节点n所有的邻近节点：
            * 如果邻近节点m在close_set中，则：
                * 跳过，选取下一个邻近节点
            * 如果邻近节点m也不在open_set中，则：
                * 设置节点m的parent为节点n
                * 计算节点m的优先级
                * 将节点m加入open_set中
            * 如果邻近节点m已经在open_set中，且新路径 $f$ 更小，则：
                * 设置节点m的parent为节点n
                * 计算节点m的优先级
```

例题4

搜索S，

发现 $f(A) = 2 + 11 = 13, f(B) = 4 + 8 = 12, f(C) = 3 + 8 = 11$，因为$11 < 12 < 13$，所以下一步搜索C，

发现 $f(H) = 6 + 5 = 11$，因为 $11 < 12 < 13$，所以下一步搜索H，

发现 $f(F) = 8 + 8 = 16$，因为 $12 < 13 < 16$，所以下一步搜索B，

发现 $f(F) = 6 + 8 = 14$，因为 $13 < 14 < 16$，所以下一步搜索A，

发现 $f(D) = 4 + 7 = 11, f(E) = 9 + 4 = 13$，因为 $11 < 13 < 14$，所以下一步搜索D

D没有可以搜索的临近点，因为 $13 < 14$，所以下一步搜索E，

发现 $f(G) = 15 + 0 = 15$，因为 $14 < 15$，所以下一步搜索F，

发现 $f(G)=7+0=7$，因为只有G还可以搜索，所以下一步搜索G，

搜索到G，搜索结束，最短路径长度为7，路径为S-B-F-G

可采纳启发函数(Admissible heuristic)

满足下列条件的启发函数可称为 admissible

$0 \leq h(n) \leq h*(n)$

h*(n)为从节点n到目标点的实际成本

例题5

显然都满足条件，所以选择ABCD

卷积(convolution)

卷积核(convolution kernel)，相当于四则运算里的加数/减数/乘数/除数

设输入矩阵 in，卷积核 K，输出矩阵 out，一般他们都是行列数相同的矩阵

$out_{size}=(in_{size}-K_{size}+2*pad)/stride+1$

边界填充(pad)默认0，步长(stride)默认1

$pad>0$，就在输入矩阵外面添上pad圈0

步长是卷积核在输入矩阵上移动的步长

当 $pad=0, stride=1$，卷积公式如下

$$out_{x,y}=\sum_{i=0}^{K_{size}}\sum_{j=0}^{K_{size}}K_{i,j}in_{x+i,y+j}$$

神经网络

由神经元组成，神经元代表最基本的分类器，例如第一个隐藏层的神经元是最简单的分类器，检测绿色、黄色、斜纹等等特征是否出现，第二个隐藏层的神经元做比第一个隐藏层更复杂的东西，根据第一层的output，如果看到直线横线，就判断是窗框的一部分，看到棕色的直线纹就判断是木纹，看到斜条纹加灰色就判断是轮胎的一部分。第三层再根据第二层的output，会做更复杂的事情，比如看到蜂巢就激活，看到车轮被激活，看到人的上半身被激活。

卷积神经网络(CNN)

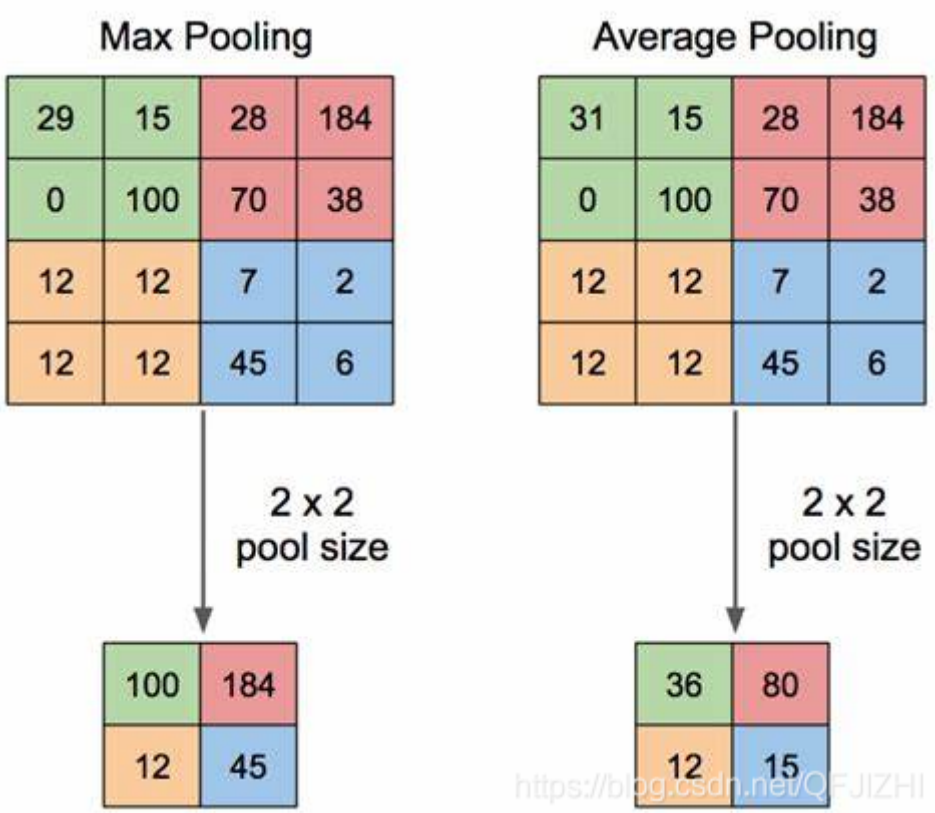
常用在图像识别，输入训练集，通过卷积层和池化层（这两层可以重复多次），再压平，然后把压平结果丢全连接神经网络，就能得到图像识别结果

相比传统神经网络，去掉一些参数，但是通过抓取特征的方式使识别更有些，例如识别鸟，可以分为识别鸟嘴，鸟爪，翅膀，尾巴等子问题，而识别鸟嘴可以简单化成识别鸟嘴的曲线，这样就不需要识别整体的鸟而是识别鸟的特征

池化层(pooling layer)

有两种池化层（最大池化，平均池化）

就是把原矩阵切块并计算一些信息形成一个新矩阵



压平(flatten)

多通道矩阵转化为一维向量，这样就能把池化层输出丢到全连接神经网络

例题6

根据上述公式计算，得

$$out = \begin{pmatrix} -1 & 6 & -2 & -2 \\ 15 & 0 & 14 & -2 \\ 6 & 13 & 18 & -4 \\ 14 & -4 & 9 & 9 \end{pmatrix}$$

最大池化：

$$= \begin{pmatrix} 15 & 14 \\ 14 & 18 \end{pmatrix}$$

聚类(clustering)

在二维坐标系上是把坐标相近的点分到一个集合，最终形成多个集合

有多种聚类方法，比如K均值(K-means)，单链路(single link)

单链路(single link)

起始每个点都是一个集合，每次选取最近的两个集合合并，两个集合距离定义为两个集合内最近点对的距离

例题7

BOS到NY的206最小，所以把这两个集合合并成BOS-NY

BOS-NY到CHI的802最小，所以把这两个集合合并成BOS-NY-CHI

SF到SEA的808最小，所以把这两个集合合并成SF-SEA

BOS-NY-CHI到DEN的966最小，所以把这两个集合合并成BOS-NY-CHI-DEN

BOS-NY-CHI-DEN到SF-SEA的1235最小，所以把这两个集合合并成
BOS-NY-CHI-DEN-SF-SEA

树状图按照上述步骤自底向上构造二叉树画

信息检索(information retrieval)

预处理

1. 爬虫收集待建索引的文档
2. 对这些文档中的文本进行词条化 (Tokenization)
3. 对步骤 2 中得到的词条 (Token) 进行语言学预处理，得到词项 (Term)
4. 根据词项对所有文档建立索引 (Index Construction)

查询

1. 系统会根据查询，为每个文档打分，这个分数就是该文档与查询的相关程度，具体分数和所用模型有关，有布尔模型和向量空间模型
2. 根据每个文档的得分进行排序
3. 返回得分最高的 k 个结果。

核化(Kernalization)

核化是计算复杂性理论领域中用于降低问题计算复杂性的技术。

它用于通过将问题实例转换为具有较少元素的等效实例来减小问题实例的大小。转换后的实例称为内核。

例如CNN用于图像识别时就是通过卷积层找出特征，来减小找到物体的难度

过拟合(Overfitting)

过拟合指的是训练的模型对训练集准确率太高而对测试集准确率太低

下列方法能对抗过拟合

1. 增加数据量：增加数据量可以提高模型的泛化能力，减少过拟合的可能性。
2. 增加正则化：正则化是一种限制模型复杂度的方法，可以减少过拟合的可能性。
3. Dropout：Dropout是一种有效的正则化技术，可以有效地减少过拟合的可能性。
4. 数据增强：数据增强是一种技术，可以通过对现有数据进行变换来增加数据量，从而减少过拟合的可能性。

5. Early Stopping：Early Stopping是一种正则化技术，可以在模型开始过拟合之前停止训练，从而减少过拟合的可能性。

K均值算法(K-means)

K均值算法是一种聚类分析算法，它将数据集中的样本点划分为K个聚类，每个聚类由一个均值向量表示，其中K是用户指定的参数。算法的基本步骤是：

1. 随机选择K个初始聚类中心；
2. 将每个样本点分配到最近的聚类中心；
3. 更新每个聚类的均值向量；
4. 重复步骤2和3，直到聚类中心不再发生变化。

布尔模型(boolean model)

布尔模型是信息检索中的一种模型，它是一种基于逻辑的模型，用于检索文档集合中的文档。它使用布尔运算符（AND，OR，NOT）来构建查询，以检索文档集合中的文档。它可以用来检索文本文档，图像，视频等多种文档类型。它的优点是可以检索出精确的文档，缺点是查询语句的构建需要较高的技术水平。

向量空间模型(vector space model)

向量空间模型是一种文本检索技术，它将文档和查询表示为向量，并使用向量空间中的距离度量来衡量文档和查询之间的相似度。VSM的基本思想是，文档和查询都可以表示为一个向量，每个向量由一组特征（即词汇）组成，每个特征都有一个权重，表示该特征在文档或查询中的重要性。VSM通过计算文档和查询之间的距离来衡量文档和查询之间的相似度，从而实现文档检索。VSM可以用来检索文本文档，也可以用来检索图像、视频等多媒体文件。

熵(entropy)

熵是系统中无序程度的度量。使用以下公式计算：

$$entropy = -\sum p \log_2 p$$

其中p是系统中每个状态的概率。

例题9

Alpha beta剪枝通过搜索所有可能的移动并使用评估函数评估每个移动来实现。

然后，它会消除不如最佳招式好的招式，并继续进行，直到找到最佳招式。重复此过程，直到找到最佳移动。

ID3

ID3算法是一种基于决策树的机器学习算法，用于从数据集中构建决策树。它使用信息增益来选择最佳分割属性，以便将数据集分割为最小的子集。它是一种贪心算法，每次选择最佳分割属性，以便最大程度地减少数据集的熵。

C4.5

C4.5算法是ID3算法的改进版本。它可以处理连续和离散属性，并且可以处理缺失值。C4.5算法的主要思想是基于信息增益率（Information Gain Ratio）来选择最优划分属性，并且可以根据需要生成不完整的决策树。

聚集式聚类(agglomerative clustering)

是一种层次聚类算法，它从每个数据点开始，然后每次将最相似的两个簇合并，直到所有的簇都被合并为一个簇为止。它是一种自下而上的聚类算法，它从每个数据点开始，然后每次将最相似的两个簇合并，直到所有的簇都被合并为一个簇为止。它可以用来发现数据集中的结构，并且可以用来检测异常值。

推理系统(reasoning systems)

1. 演绎：演绎推理是从一般原理出发，推导出特定的结论。它是从一般原理出发，推导出特定的结论，而不是从特定的结论出发，推导出一般原理。
2. 归纳：归纳推理是从特定的结论出发，推导出一般原理。它是从特定的结论出发，推导出一般原理，而不是从一般原理出发，推导出特定的结论。
3. 溯因：溯因推理是从一个结果出发，推导出它的原因。它是从一个结果出发，推导出它的原因，而不是从一般原理出发，推导出特定的结论或从特定的结论出发，推导出一般原理。

例题13

卷积神经网络（CNN）使用卷积层来提取图像特征，以便进行分类。

全卷积神经网络（FCNN）使用全卷积层来提取图像特征，以便进行分割。

与CNN不同，FCNN使用全卷积层，这意味着它可以提取更多的特征，并且可以处理更大的图像。

情感分析(sentimental analysis)

自然语言处理（NLP）方法的应用，它是对带有情感色彩的主观性文本进行分析、处理、归纳和推理，利用一些情感得分指标来量化定性数据的方法

下面是其步骤

1. 数据收集：从各种来源收集数据
2. 数据预处理：通过去除噪声、停止词、标点符号等来清理数据。
3. 特征提取：从数据中提取特征，如单词计数、n-gram等。
4. 模型训练：使用监督学习算法（如朴素贝叶斯、支持向量机等）在提取的特征上训练模型。
5. 模型评估：使用准确性、准确性、召回率等指标评估模型的性能。
6. 模型部署：在生产环境中部署模型以进行实时情绪分析。

搜索引擎(search engine)

搜索引擎的工作原理是使用网络爬虫扫描网络以查找内容。爬行器收集有关它找到的内容的信息，例如使用的单词、到其他页面的链接以及页面的结构。然后将这些信息存储在一个索引中，该索引是爬行器找到的所有内容的数据库。

当用户搜索某项内容时，搜索引擎会通过索引查找最相关的结果。它通过使用算法来确定哪些页面最可能包含用户正在查找的信息。然后，搜索引擎根据相关性对结果进行排序，并将其显示给用户。

KNN

KNN算法是一种基于实例的学习算法，它的工作原理是：给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的k个实例，这k个实例的多数属于某个类，就把该输入实例分为这个类。KNN算法是一种简单的分类算法，它不需要做任何的训练，而是直接根据训练数据集来进行分类。